arXiv:1809.02736v1 [cs.CV] 8 Sep 2018

Joint Autoregressive and Hierarchical Priors for Learned Image Compression

David Minnen, Johannes Ballé, George Toderici
Google Research
{dminnen, jballe, gtoderici}@google.com

Abstract

Recent models for learned image compression are based on autoencoders, learning approximately invertible mappings from pixels to a quantized latent representation. These are combined with an entropy model, a prior on the latent representation that can be used with standard arithmetic coding algorithms to yield a compressed bitstream. Recently, hierarchical entropy models have been introduced as a way to exploit more structure in the latents than simple fully factorized priors, improving compression performance while maintaining end-to-end optimization. Inspired by the success of autoregressive priors in probabilistic generative models, we examine autoregressive, hierarchical, as well as combined priors as alternatives, weighing their costs and benefits in the context of image compression. While it is well known that autoregressive models come with a significant computational penalty, we find that in terms of compression performance, autoregressive and hierarchical priors are complementary and, together, exploit the probabilistic structure in the latents better than all previous learned models. The combined model yields state-of-the-art rate–distortion performance, providing a 15.8% average reduction in file size over the previous state-of-the-art method based on deep learning, which corresponds to a 59.8% size reduction over JPEG, more than 35% reduction compared to WebP and JPEG2000, and bitstreams 8.4% smaller than BPG, the current state-of-the-art image codec. To the best of our knowledge, our model is the first learning-based method to outperform BPG on both PSNR and MS-SSIM distortion metrics.

1 Introduction

Most recent methods for learning-based, lossy image compression adopt an approach based on *transform coding* [1]. In this approach, image compression is achieved by first mapping pixel data into a quantized latent representation and then losslessly compressing the latents. Within the deep learning research community, the transforms typically take the form of convolutional neural networks (CNNs), which approximate nonlinear functions with the potential to map pixels into a more compressible latent space than the linear transforms used by traditional image codecs. This nonlinear transform coding method resembles an autoencoder [2], [3], which consists of an *encoder* transform between the data (in this case, pixels) and a latent, reduced-dimensionality space, and a *decoder*, an approximate inverse function that maps latents back to pixels. While dimensionality reduction can be seen as a simplistic form of compression, it is not equivalent to it, as the goal of compression is to reduce the entropy of the representation under a prior probability model shared between the sender and the receiver (the entropy model), not only the dimensionality. To improve compression performance, recent methods have given increased focus to this part of the model [4]–[14]. Finally, the entropy model is used in conjunction with standard entropy coding algorithms such as arithmetic, range, or Huffman coding [15]–[17] to generate a compressed bitstream.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

The training goal is to minimize the expected length of the bitstream as well as the expected distortion of the reconstructed image with respect to the original, giving rise to a rate–distortion optimization problem:

$$R + \lambda \cdot D = \underbrace{\mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{x}}} \left[-\log_2 p_{\hat{\boldsymbol{y}}}(\lfloor f(\boldsymbol{x}) \rceil) \right]}_{\text{rate}} + \lambda \cdot \underbrace{\mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{x}}} \left[d(\boldsymbol{x}, g(\lfloor f(\boldsymbol{x}) \rceil)) \right]}_{\text{distortion}}, \tag{1}$$

where λ is the Lagrange multiplier that determines the desired rate–distortion trade-off, $p_{\boldsymbol{x}}$ is the unknown distribution of natural images, $\lfloor \cdot \rceil$ represents rounding to the nearest integer (quantization), $\boldsymbol{y} = f(\boldsymbol{x})$ is the encoder, $\hat{\boldsymbol{y}} = \lfloor \boldsymbol{y} \rceil$ are the quantized latents, $p_{\hat{\boldsymbol{y}}}$ is a discrete entropy model, and $\hat{\boldsymbol{x}} = g(\hat{\boldsymbol{y}})$ is the decoder with $\hat{\boldsymbol{x}}$ representing the reconstructed image. The rate term corresponds to the cross entropy between the marginal distribution of the latents and the learned entropy model, which is minimized when the two distributions are identical. The distortion term may correspond to a closed-form likelihood, such as when $d(\boldsymbol{x}, \hat{\boldsymbol{x}})$ represents mean squared error (MSE), which induces an interpretation of the model as a variational autoencoder [6]. When optimizing the model for other distortion metrics such as MS-SSIM, it is simply minimized as an energy function.

The models we analyze in this paper build on the work of Ballé et al. [13], which uses a noise-based relaxation to be able to apply gradient descent methods to the loss function in Eq. (1) and introduces a hierarchical prior to improve the entropy model. While most previous research uses a fixed, though potentially complex, entropy model, Ballé et al. use a Gaussian scale mixture (GSM) [18] where the scale parameters are conditioned on a hyperprior. Their model allows for end-to-end training, which includes joint optimization of a quantized representation of the hyperprior, the conditional entropy model, and the base autoencoder. The key insight is that the compressed hyperprior could be added to the generated bitstream as *side information*, which allows the decoder to use the conditional entropy model. In this way, the entropy model itself is image-dependent and spatially adaptive, which allows for a richer and more accurate model. Ballé et al. show that standard optimization methods for deep neural networks are sufficient to learn a useful balance between the size of the side information and the savings gained from a more accurate entropy model. The resulting compression model provides state-of-the-art image compression results compared to earlier learning-based methods.

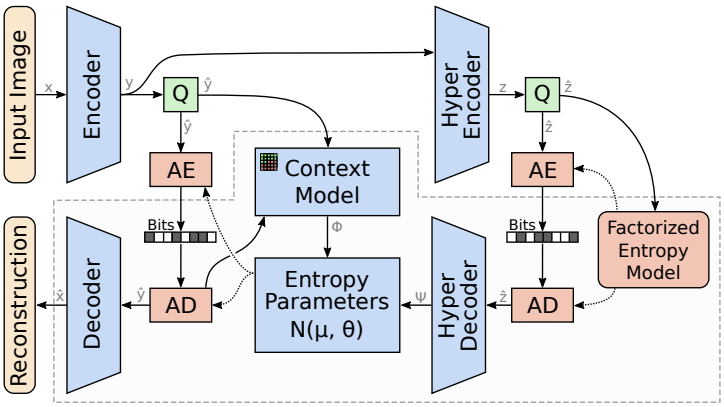
We extend this GSM-based entropy model in two ways: first, by generalizing the hierarchical GSM model to a Gaussian mixture model, and, inspired by recent work on generative models, by adding an autoregressive component. We assess the compression performance of both approaches, including variations in the network architectures, and discuss benefits and potential drawbacks of both extensions. For the results in this paper, we did not make efforts to reduce the capacity (i.e., number of channels, layers) of the artificial neural networks to optimize computational complexity, since we are interested in determining the potential of different forms of priors rather than trading off complexity against performance. Note that increasing capacity alone is not sufficient to obtain arbitrarily good compression performance [13, appendix 6.3].

2 Architecture Details

Figure 1 provides a high-level overview of our generalized compression model, which contains two main sub-networks[1]. The first is the core autoencoder, which learns a quantized latent representation of images (*Encoder* and *Decoder* blocks). The second sub-network is responsible for learning a probabilistic model over quantized latents used for entropy coding. It combines the *Context Model*, an autoregressive model over latents, with the hyper-network (*Hyper Encoder* and *Hyper Decoder* blocks), which learns to represent information useful for correcting the context-based predictions. The data from these two sources is combined by the *Entropy Parameters* network, which generates the mean and scale parameters for a conditional Gaussian entropy model.

Once training is complete, a valid compression model must prevent any information from passing between the encoder to the decoder unless that information is available in the compressed file. In Figure 1, the arithmetic encoding (AE) blocks produce the compressed representation of the symbols coming from the quantizer, which is stored in a file. Therefore at decoding time, any information that depends on the quantized latents may be used by the decoder once it has been decoded. In order for the context model to work, at any point it can only access the latents that have already been decoded.

[1] See Section 4 in the supplemental materials for an in-depth visual comparison between our architecture variants and previous learning-based methods.



Component	Symbol
Input Image	$\boldsymbol{x}$
Encoder	$f(\boldsymbol{x};\boldsymbol{\theta}_e)$
Latents	$\boldsymbol{y}$
Latents (quantized)	$\hat{\boldsymbol{y}}$
Decoder	$g(\hat{\boldsymbol{y}};\boldsymbol{\theta}_d)$
Hyper Encoder	$f_h(\boldsymbol{y};\boldsymbol{\theta}_{he})$
Hyper-latents	$\boldsymbol{z}$
Hyper-latents (quant.)	$\hat{\boldsymbol{z}}$
Hyper Decoder	$g_h(\hat{\boldsymbol{z}};\boldsymbol{\theta}_{hd})$
Context Model	$g_{cm}(\boldsymbol{y}_{<i};\boldsymbol{\theta}_{cm})$
Entropy Parameters	$g_{ep}(\cdot;\boldsymbol{\theta}_{ep})$
Reconstruction	$\hat{\boldsymbol{x}}$

Figure 1: Our combined model jointly optimizes an autoregressive component that predicts latents from their causal context (*Context Model*) along with a hyperprior and the underlying autoencoder. Real-valued latent representations are quantized (*Q*) to create latents ($\hat{\boldsymbol{y}}$) and hyper-latents ($\hat{\boldsymbol{z}}$), which are compressed into a bitstream using an arithmetic encoder (*AE*) and decompressed by an arithmetic decoder (*AD*). The highlighted region corresponds to the components that are executed by the receiver to recover an image from a compressed bitstream.

When starting to decode an image, we assume that the previously decoded latents have all been set to zero.

The learning problem is to minimize the expected rate–distortion loss defined in Eq. 1 over the model parameters. Following the work of Ballé et al. [13], we model each latent, $\hat{y}_i$, as a Gaussian convolved with a unit uniform distribution. This ensures a good match between encoder and decoder distributions of both the quantized latents, and continuous-valued latents subjected to additive uniform noise during training. While [13] predicted the scale of each Gaussian conditioned on the hyperprior, $\hat{\boldsymbol{z}}$, we extend the model by predicting the mean and scale parameters conditioned on both the hyperprior as well as the causal context of each latent $\hat{y}_i$, which we denote $\hat{\boldsymbol{y}}_{<i}$. The predicted Gaussian parameters are functions of the learned parameters of the hyper-decoder, context model, and entropy parameters networks ($\boldsymbol{\theta}_{hd}$, $\boldsymbol{\theta}_{cm}$, and $\boldsymbol{\theta}_{ep}$, respectively):

$$p_{\hat{\boldsymbol{y}}}(\hat{\boldsymbol{y}} \mid \hat{\boldsymbol{z}}, \boldsymbol{\theta}_{hd}, \boldsymbol{\theta}_{cm}, \boldsymbol{\theta}_{ep}) = \prod_i \left(\mathcal{N}(\mu_i, \sigma_i^2) * \mathcal{U}(-\tfrac{1}{2}, \tfrac{1}{2}) \right)(\hat{y}_i)$$
$$\text{with } \mu_i, \sigma_i = g_{ep}(\boldsymbol{\psi}, \boldsymbol{\phi}_i; \boldsymbol{\theta}_{ep}), \boldsymbol{\psi} = g_h(\hat{\boldsymbol{z}}; \boldsymbol{\theta}_{hd}), \text{and } \boldsymbol{\phi}_i = g_{cm}(\hat{\boldsymbol{y}}_{<i}; \boldsymbol{\theta}_{cm}). \quad (2)$$

The entropy model for the hyperprior is the same as in [13], although we expect the hyper-encoder and hyper-decoder to learn significantly different functions in our combined model, since they now work in conjunction with an autoregressive network to predict the entropy model parameters. Since we do not make any assumptions about the distribution of the hyper-latents, a non-parametric, fully factorized density model is used. A more powerful entropy model for the hyper-latents may improve compression rates, e.g., we could stack multiple instances of our contextual model, but we expect the net effect to be minimal since $\hat{\boldsymbol{z}}$ comprises only a very small percentage of the total file size. Because both the compressed latents and the compressed hyper-latents are part of the generated bitstream, the rate–distortion loss from Equation 1 must be expanded to include the cost of transmitting $\hat{\boldsymbol{z}}$. Coupled with a squared error distortion metric, the full loss function becomes:

$$R + \lambda \cdot D = \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}}\left[-\log_2 p_{\hat{\boldsymbol{y}}}(\hat{\boldsymbol{y}})\right]}_{\text{rate (latents)}} + \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}}\left[-\log_2 p_{\hat{\boldsymbol{z}}}(\hat{\boldsymbol{z}})\right]}_{\text{rate (hyper-latents)}} + \lambda \cdot \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}}\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2}_{\text{distortion}} \quad (3)$$

2.1 Layer Details and Constraints

Details about the individual network layers in each component of our models are outlined in Table 1. While the internal structure of the components is fairly unrestricted, e.g., one could exchange the convolutional layers for residual blocks or dilated convolution without fundamentally changing the model, certain components must be constrained to ensure that availability of the bitstreams alone is sufficient for the receiver to reconstruct the image.

Encoder	Decoder	Hyper Encoder	Hyper Decoder	Context Prediction	Entropy Parameters
Conv: 5×5 c192 s2	Deconv: 5×5 c192 s2	Conv: 3×3 c192 s1	Deconv: 5×5 c192 s2	Masked: 5×5 c384 s1	Conv: 1×1 c640 s1
GDN	IGDN	Leaky ReLU	Leaky ReLU		Leaky ReLU
Conv: 5×5 c192 s2	Deconv: 5×5 c192 s2	Conv: 5×5 c192 s2	Deconv: 5×5 c288 s2		Conv: 1×1 c512 s1
GDN	IGDN	Leaky ReLU	Leaky ReLU		Leaky ReLU
Conv: 5×5 c192 s2	Deconv: 5×5 c192 s2	Conv: 5×5 c192 s2	Deconv: 3×3 c384 s1		Conv: 1×1 c384 s1
GDN	IGDN				
Conv: 5×5 c192 s2	Deconv: 5×5 c3 s2				

Table 1: Each row corresponds to a layer of our generalized model. Convolutional layers are specified with the "Conv" prefix followed by the kernel size, number of channels and downsampling stride (e.g., the first layer of the encoder uses 5×5 kernels with 192 channels and a stride of two). The "Deconv" prefix corresponds to upsampled convolutions (i.e., in TensorFlow, `tf.conv2d_transpose`), while "Masked" corresponds to masked convolution as in [19]. GDN stands for generalized divisive normalization, and IGDN is inverse GDN [20].

The last layer of the encoder corresponds to the bottleneck of the base autoencoder. Its number of output channels determines the number of elements that must be compressed and stored. Depending on the rate–distortion trade-off, our models learn to ignore certain channels by deterministically generating the same latent value and assigning it a probability of 1, which wastes computation but requires no additional entropy. This modeling flexibility allows us to set the bottleneck larger than necessary, and let the model determine the number of channels that yield the best performance. Similar to reported in other work, we found that too few channels in the bottleneck can impede rate–distortion performance when training models to target higher bit rates, but too having too many does not harm the compression performance.

The final layer of the decoder must have three channels to generate RGB images, and the final layer of the *Entropy Parameters* sub-network must have exactly twice as many channels as the bottleneck. This constraint arises because the *Entropy Parameters* network predicts two values, the mean and scale of a Gaussian distribution, for each latent. The number of output channels of the *Context Model* and *Hyper Decoder* components are not constrained, but we also set them to twice the bottleneck size in all of our experiments.

Although the formal definition of our model allows the autoregressive component to condition its predictions $\boldsymbol{\phi}_i = g_{cm}(\hat{\boldsymbol{y}}_{<i}; \boldsymbol{\theta}_{cm})$ on all previous latents, in practice we use a limited context (5×5 convolution kernels) with masked convolution similar to the approach used by PixelCNN [19]. The *Entropy Parameters* network is also constrained, since it can not access predictions from the *Context Model* beyond the current latent element. For simplicity, we use 1×1 convolution in the *Entropy Parameters* network, although masked convolution is also permissible. Section 3 provides an empirical evaluation of the model variants we assessed, exploring the effects of different context sizes and more complex autoregressive networks.

3 Experimental Results

We evaluate our generalized models by calculating the rate–distortion (RD) performance averaged over the publicly available Kodak image set [21][2]. Figure 2 shows RD curves using peak signal-to-noise ratio (PSNR) as the image quality metric. While PSNR is known to be a relatively poor perceptual metric [22], it is still a standard metric used to evaluate image compression algorithms and is the primary metric used for tuning conventional compression methods. The RD graph on the left of Figure 2 compares our combined context + hyperprior model to existing image codecs (standard codecs and learned models) and shows that this model outperforms all of the existing methods including BPG [23], a state-of-the-art codec based on the intra-frame coding algorithm from HEVC [24]. To the best of our knowledge, it is the first learning-based compression model to outperform BPG on PSNR. The right RD graph compares different versions of our models and shows that the combined model performs the best, while the context-only model performs slightly worse than either hierachical version.

[2]Please see the supplemental material for additional evaluation results including full-page RD curves, example images, and results on the larger Tecnick image set (100 images with resolution 1200×1200).

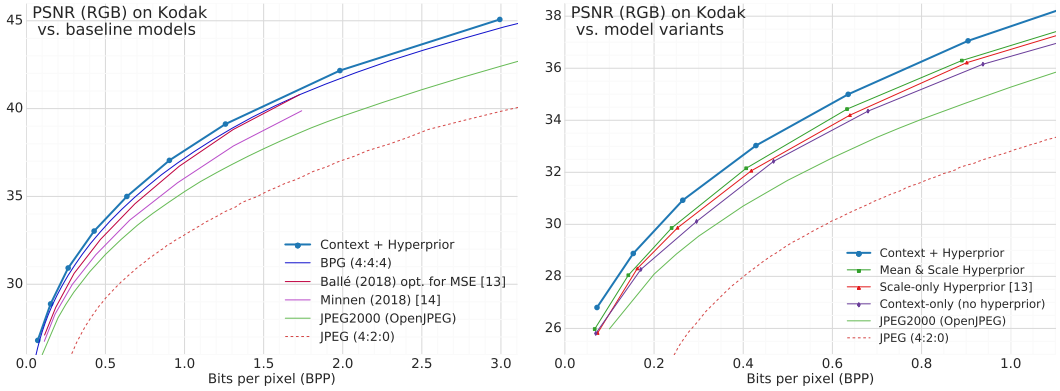


Figure 2: Our combined approach (context + hyperprior) has better rate–distortion performance on the Kodak image set as measured by PSNR (RGB) compared to all of the baselines methods (*left*). To our knowledge, this is the first learning-based method to outperform BPG on PSNR. The right graph compares the relative performance of different versions of our method. It shows that using a hyperprior is better than a purely autoregressive (context-only) approach and that combining both (context + hyperprior) yields the best RD performance.

Figure 3 shows RD curves for Kodak using multiscale structural similarity (MS-SSIM) [25] as the image quality metric. The graph includes two versions of our combined model: one optimized for MSE and one optimized for MS-SSIM. The latter outperforms all existing methods including all standard codecs and other learning-based methods that were also optimized for MS-SSIM ([6], [9], [13]). As expected, when our model is optimized for MSE, performance according to MS-SSIM falls. Nonetheless, the MS-SSIM scores for this model still exceed all standard codecs and all learning-based methods that were not specifically optimized for MS-SSIM.

As outlined in Table 1, our baseline architecture for the combined model uses 5×5 masked convolution in a single linear layer for the context model, and it uses a conditional Gaussian distribution for the entropy model. Figure 4 compares this baseline to several variants by showing the relative increase in file size at a single rate-point. The green bars show that exchanging the Gaussian distribution for a logistic distribution has almost no effect (the 0.3% increase is smaller than the training variance), while switching to a Laplacian distribution decreases performance more substantially. The blue bars compare different context configurations. Masked 3×3 and 7×7 convolution both perform slightly worse, which is surprising since we expected the additional context provided by the 7×7 kernels to improve prediction accuracy. Similarly, a 3-layer, nonlinear context model using 5×5 masked convolution also performed slightly worse than the linear baseline. Finally, the purple bars show the effect of using a severely restricted context such as only a single neighbor or three neighbors from the previous row. The primary benefit of these models is increased parallelization when calculating context-based predictions since the dependence is reduced from two dimensions down to one. While both cases show a non-negligible rate increase (2.1% and 3.1%, respectively), the increase may be worthwhile in a practical implementation where runtime speed is a major concern.

Finally, Figure 5 provides a visual comparison for one of the Kodak images. Creating accurate comparisons is difficult since most compression methods do not have the ability to target a precise bit rate. We therefore selected comparison images with sizes that are as close as possible, but always larger than our encoding (up to 9.4% larger in the case of BPG). Nonetheless, our compression model provides clearly better visual quality compared to the scale hyperprior baseline [13] and JPEG. The perceptual quality relative to BPG is much closer. For example, BPG preserves mode detail in the sky and parts of the fence, but at the expense of introducing geometric artifacts in the sky, mild ringing near the building/sky boundaries, and some boundary artifacts where neighboring blocks have widely different levels of detail (e.g., in the grass and lighthouse).

4 Related Work

The earliest research that used neural networks to compress images dates back to the 1980s and relies on an autoencoder with a small bottleneck using either uniform quantization [26] or vector

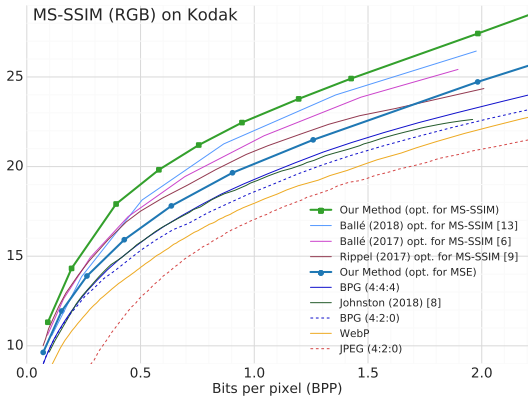


Figure 3: When evaluated using MS-SSIM (RGB) on Kodak, our combined approach has better RD performance than all previous methods when optimized for MS-SSIM. When optimized for MSE, our method still provides better MS-SSIM scores than all of the standard codecs.

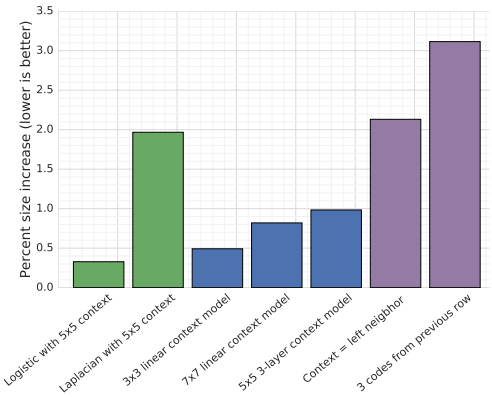
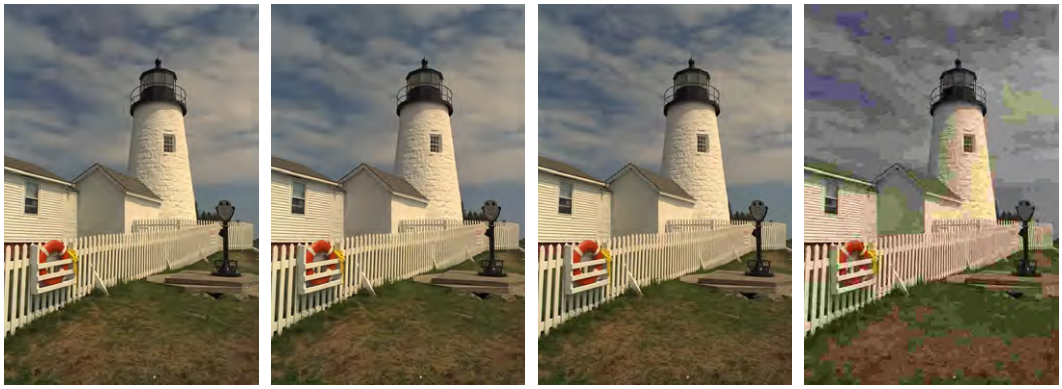


Figure 4: The baseline implementation of our model uses a hyperprior and a linear context model with 5×5 masked convolution. Optimized with $\lambda = 0.025$ (bpp ≈ 0.61 on Kodak), the baseline outperforms the other variants we tested (see text for details).

quantization [27], [28]. These approaches sought equal utilization of the codes and thus did not learn an explicit entropy model. Considerable research followed these initial models, and Jiang provides a comprehensive survey covering methods published through the late 1990s [29].

More recently, image compression with deep neural networks became a popular research topic starting with the work of Toderici et al. [30] who used a recurrent architecture based on LSTMs to learn multi-rate, progressive models. Their approach was improved by exploring other recurrent architectures for the autoencoder, training an LSTM-based entropy model, and adding a post-process that spatially adapts the bit rate based on the complexity of the local image content [4], [8]. Related research followed a more traditional image coding approach and explicitly divided images into patches instead of using a fully convolutional model [10], [31]. Inspired by modern image codecs and learned inpainting algorithms, these methods trained a neural network to predict each image patch from its causal context (in the image space, not the latent space) before encoding the residual.



(a) Ours (0.2149 bpp) (b) Scale-only (0.2205 bpp) (c) BPG (0.2352 bpp) (d) JPEG (0.2309 bpp)

Figure 5: At similar bit rates, our combined method provides the highest visual quality. Note the aliasing in the fence in the scale-only version as well as a slight global color cast and blurriness in the yellow rope. BPG shows more "classical" compression artifacts, e.g., ringing around the top of the lighthouse and the roof of the middle building. BPG also introduces a few geometric artifacts in the sky, though it does preserve extra detail in the sky and fence (albeit with 9.4% more bits that our encoding). JPEG shows severe blocking artifacts at this bit rate.

Similarly, most modern image compression standards use context to predict pixel values as well as using a context-adaptive entropy model [23], [32], [33].

Many learning-based methods take the form of an autoencoder, and multiple models are trained to target different bit rates instead of training a single recurrent model [5]–[7], [9], [11], [12], [14]. Some use a fully factorized entropy model [5], [6], while others make use of context in code space to improve compression rates [4], [7]–[9], [12]. Other methods do not make use of context via an autoregressive model and instead rely on side information that is either predicted by a neural network [13] or composed of indices into a (shared) dictionary of non-parametric code distributions used locally by the entropy coder [14].

Learned image compression is also related to Bayesian generative models such as PixelCNN [19], variational autoencoders [34], PixelVAE [35], β-VAE [36], and VLAE [37]. In general, Bayesian image models seek to maximize the *evidence* $\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}} \log p(\boldsymbol{x})$, which is generally intractable, and use the joint likelihood, as in Eq. (1), as a lower bound, while compression models directly seek to optimize Eq. (1). Research on generative models is typically interested in uncovering semantically meaningful, disentangled latent representations. It has been noted that under certain conditions, compression models are formally equivalent to VAEs [5], [6]. β-VAEs have a particularly strong connection since β controls the trade-off between the data log-likelihood (distortion) and prior (rate), as does λ in our formulation, which is derived from classical rate–distortion theory.

A further major difference are the constraints imposed on compression models by the need to quantize and arithmetically encode the latents, which require certain choices regarding the parametric form of the densities and a transition between continous (differential) and discrete (Shannon) entropies. We can draw strong conceptual parallels between our models and PixelCNN autoencoders [19], and especially PixelVAE [35] and VLAE [37], when applied to discrete latents. These models are often evaluated by comparing average likelihoods (which correspond to differential entropies), whereas compression models are typically evaluated by comparing several bit rates (corresponding to Shannon entropies) and distortion values across the rate–distortion frontier, making evaluations more complex.

5 Discussion

Our approach extends the work of Ballé et al. [13] in two ways. First, we generalize the GSM model to a conditional Gaussian mixture model (GMM). Supporting this model is simply a matter of generating both a mean and a scale parameter conditioned on the hyperprior. Intuitively, the average likelihood of the observed latents increases when the center of the conditional Gaussian is closer to the true value and a smaller scale is predicted, i.e., more structure can be exploited by modeling conditional means. The core question is whether or not the benefits of this more sophisticated model outweigh the cost of the associated side information. We showed in Figure 2 (*right*) that a GMM-based entropy model provides a net benefit and outperforms the simpler GSM-based model in terms of rate–distortion performance without increasing the asymptotic complexity of the model.

The second extension that we explore is the idea of combining an autoregressive model with the hyperprior. Intuitively, we can see how these components are complementary in two ways. First, starting from the perspective of the hyperprior, we see that for identical hyper-network architectures, improvements to the entropy model require more side information. The side information increases the total compressed file size, which limits its benefit. In contrast, introducing an autoregressive component into the prior does not incur a potential rate penalty since the predictions are based only on the causal context, i.e., on latents that have already been decoded. Similarly, from the perspective of the autoregressive model, we expect some amount of uncertainty that can not be eliminated solely from the causal context. The hyperprior, however, can "look into the future" since it is part of the compressed bitstream and is fully known by the decoder. The hyperprior can thus learn to store information needed to reduce the uncertainty in the autoregressive model while avoiding information that can be accurately predicted from context.

Figure 6 visualizes some of the internal mechanisms of our models. We show three of the variants: one Gaussian scale mixture equivalent to [13], another strictly hierarchical prior extended to a Gaussian mixture model, and one combined model using an autoregressive component and a hyperprior. After encoding the lighthouse image shown in Figure 5, we extracted the latents for the channel with the highest entropy. These latents are visualized in the first column of Figure 6. The second column holds the conditional means and clearly shows the added detail attained with an autoregressive component,

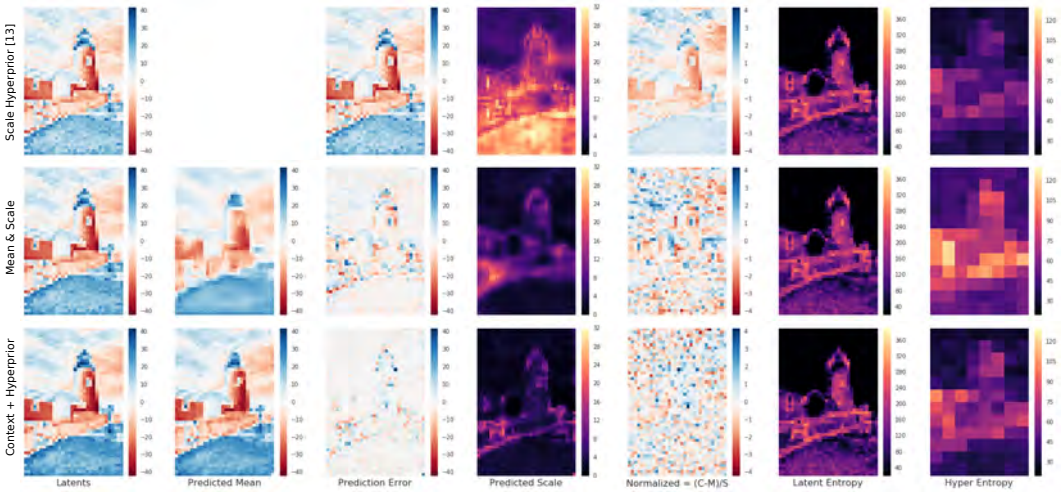


Figure 6: Each row corresponds to a different model variant and shows information for the channel with the highest entropy. The visualizations show that more powerful models reduce the prediction error, require smaller scale parameters, and remove structure from the normalized latents, which directly translates into a more accurate entropy model and thus higher compression rates.

which is reminiscent of the observation that VAE-based models tend to produce blurrier images than autoregressive models [35]. This improvement leads to a lower prediction error (third column) and smaller predicted scales, i.e. smaller uncertainty (fourth column). Our entropy model assumes that latents are conditionally independent given the hyperprior, which implies that the normalized latents, i.e. values with the predicted mean and scale removed, should be closer to i.i.d. Gaussian noise. The fifth column of Figure 6 shows that the combined model is closest to this ideal and that the autoregressive model helps significantly (compare row 4 with row 2). Finally, the last two columns show how the entropy is distributed across the image for the latents and hyper-latents.

From a practical standpoint, autoregressive models are less desirable than hierarchical models since they are inherently serial, and therefore can not be sped up using techniques such as parallelization. To report the performance of the compression models which contain an autoregressive component, we refrained from implementing a full decoder for this paper, and instead compare Shannon entropies. We have empirically verified that these measurements are within a fraction of a percent of the size of the bitstream generated by arithmetic coding.

Probability density distillation has been successfully used to get around the serial nature of autoregressive models for the task of speech synthesis [38], but unfortunately the same type of method cannot be applied in the domain of compression due to the coupling between the prior and the arithmetic decoder. To address these computational concerns, we have begun to explore very lightweight context models as described in Section 3 and Figure 4, and are considering further techniques to reduce the computational requirements of the *Context Model* and *Entropy Parameters* networks, such as engineering a tight integration of the arithmetic decoder with a differentiable autoregressive model. An alternative direction for future research may be to avoid the causality issue altogether by introducing yet more complexity into strictly hierarchical priors.

References

[1] V. K. Goyal, "Theoretical foundations of transform coding," *IEEE Signal Processing Magazine*, vol. 18, no. 5, 2001. DOI: `10.1109/79.952802`.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," in, D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., Cambridge, MA, USA: MIT Press, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362, ISBN: 0-262-68053-X. [Online]. Available: `http://dl.acm.org/citation.cfm?id=104279.104293`.

[3] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006. DOI: 10.1126/science.1127647.

[4] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. DOI: 10.1109/CVPR.2017.577. arXiv: 1608.05148.

[5] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," 2017, presented at the 5th Int. Conf. on Learning Representations.

[6] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *ArXiv e-prints*, 2017, presented at the 5th Int. Conf. on Learning Representations. arXiv: 1611.01704.

[7] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning convolutional networks for content-weighted image compression," *ArXiv e-prints*, 2017. arXiv: 1703.10553.

[8] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. J. Hwang, J. Shor, and G. Toderici, "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks," in *2018 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[9] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *Proc. of Machine Learning Research*, vol. 70, 2017, pp. 2922–2930.

[10] D. Minnen, G. Toderici, M. Covell, T. Chinen, N. Johnston, J. Shor, S. J. Hwang, D. Vincent, and S. Singh, "Spatially adaptive image compression using a tiled deep network," *International Conference on Image Processing*, 2017.

[11] E. Þ. Ágústsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 1141–1151.

[12] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, "Conditional probability models for deep image compression," in *2018 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[13] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," *6th Int. Conf. on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rkcQFMZRb.

[14] D. Minnen, G. Toderici, S. Singh, S. J. Hwang, and M. Covell, "Image-dependent local entropy models for image compression with deep networks," *International Conference on Image Processing*, 2018.

[15] J. Rissanen and G. G. Langdon Jr., "Universal modeling and coding," *IEEE Transactions on Information Theory*, vol. 27, no. 1, 1981. DOI: 10.1109/TIT.1981.1056282.

[16] G. Martin, "Range encoding: An algorithm for removing redundancy from a digitized message," in *Video & Data Recording Conference*, Jul. 1979.

[17] J. van Leeuwen, "On the construction of huffman trees," in *ICALP*, 1976, pp. 382–410.

[18] M. J. Wainwright and E. P. Simoncelli, "Scale mixtures of gaussians and the statistics of natural images," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99, Denver, CO: MIT Press, 1999, pp. 855–861.

[19] A. van den Oord, N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals, and A. Graves, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., Curran Associates, Inc., 2016, pp. 4790–4798.

[20] J. Ballé, V. Laparra, and E. P. Simoncelli, "Density modeling of images using a generalized normalization transformation," *ArXiv e-prints*, 2016, presented at the 4th Int. Conf. on Learning Representations. arXiv: 1511.06281.

[21] E. Kodak, *Kodak lossless true color image suite (PhotoCD PCD0992)*. [Online]. Available: http://r0k.us/graphics/kodak/.

[22] N. Ponomarenko, L. Jin, O. Ieremeiev, V. Lukin, K. Egiazarian, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, and C.-C. Jay Kuo, "Image database TID2013," *Image Commun.*, vol. 30, no. C, pp. 57–77, Jan. 2015, ISSN: 0923-5965. DOI: 10.1016/j.image.2014.10.009.

[23] F. Bellard, *BPG image format (http://bellard.org/bpg/)*, Accessed: 2017-01-30. [Online]. Available: http://bellard.org/bpg/.

[24] *ITU-R rec. H.265 & ISO/IEC 23008-2: High efficiency video coding*, 2013.

[25] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, IEEE, vol. 2, 2003, pp. 1398–1402.

[26] G. W. Cottrell, P. Munro, and D. Zipser, "Image compression by back propagation: An example of extensional programming," in *Models of Cognition: A Review of Cognitive Science*, N. E. Sharkey, Ed., Also presented at the Ninth Ann Meeting of the Cognitive Science Society, 1987, pp. 461-473, vol. 1, Norwood, NJ, 1989.

[27] S. Luttrell, "Image compression using a neural network," in *Pattern Recognition Letters*, vol. 10, Oct. 1988, pp. 1231–1238.

[28] E. Watkins Bruce, *Data compression using artificial neural networks.* 1991. [Online]. Available: `https://calhoun.nps.edu/handle/10945/25801`.

[29] J. Jiang, "Image compression with neural networks–a survey," *Signal Processing: Image Communication*, vol. 14, pp. 737–760, 1999.

[30] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," *ArXiv e-prints*, 2016, presented at the 4th Int. Conf. on Learning Representations. arXiv: `1511.06085`.

[31] M. H. Baig, V. Koltun, and L. Torresani, "Learning to inpaint for image compression," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 1246–1255.

[32] "Information technology–JPEG 2000 image coding system," International Organization for Standardization, Geneva, CH, Standard, Dec. 2000.

[33] Google, *WebP: Compression techniques*, Accessed: 2017-01-30. [Online]. Available: `http://developers.google.com/speed/webp/docs/compression`.

[34] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *ArXiv e-prints*, 2014, Presented at the 2nd Int. Conf. on Learning Representations. arXiv: `1312.6114`.

[35] I. Gulrajani, K. Kumar, F. Ahmed, A. Ali Taiga, F. Visin, D. Vazquez, and A. Courville, "PixelVAE: A latent variable model for natural images," 2017, presented at the 5th Int. Conf. on Learning Representations.

[36] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-VAE: Learning basic visual concepts with a constrained variational framework," 2017, presented at the 5th Int. Conf. on Learning Representations.

[37] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel, "Variational lossy autoencoder," 2017, presented at the 5th Int. Conf. on Learning Representations.

[38] A. v. d. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, *et al.*, "Parallel wavenet: Fast high-fidelity speech synthesis," *ArXiv preprint arXiv:1711.10433*, 2017.

Appendix A Rate-Distortion Curves

A.1 PSNR on Kodak

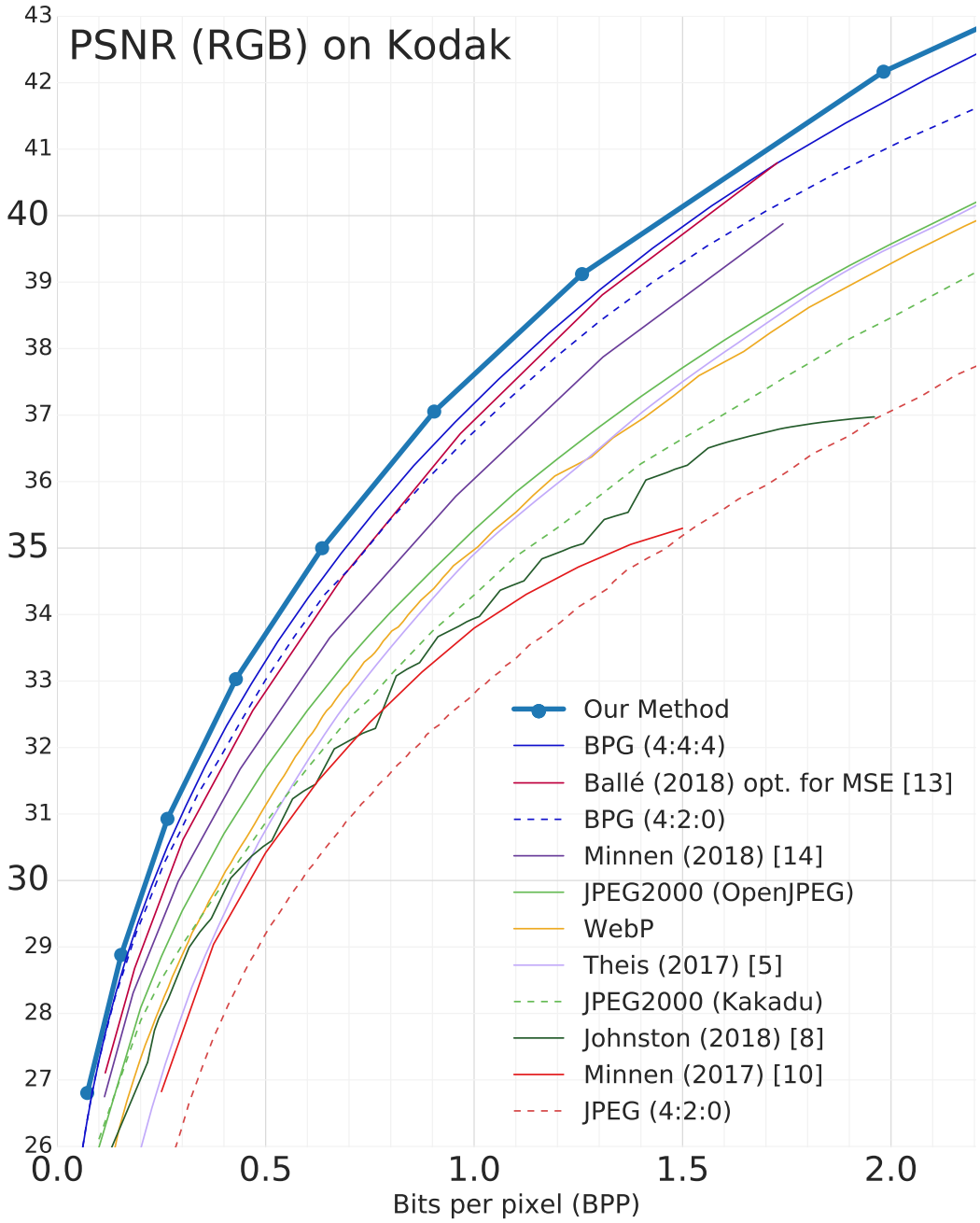


Figure 7: When trained for MSE, our method provides better rate-distortion performance compared to all of the baseline methods when evaluated on the Kodak image set using PSNR. Each point on the RD curves is calculated by averaging over the PSNR and bit rate for the 24 Kodak images for a single Q value (for standard codecs) or λ (for learned methods).

A.2 MS-SSIM on Kodak

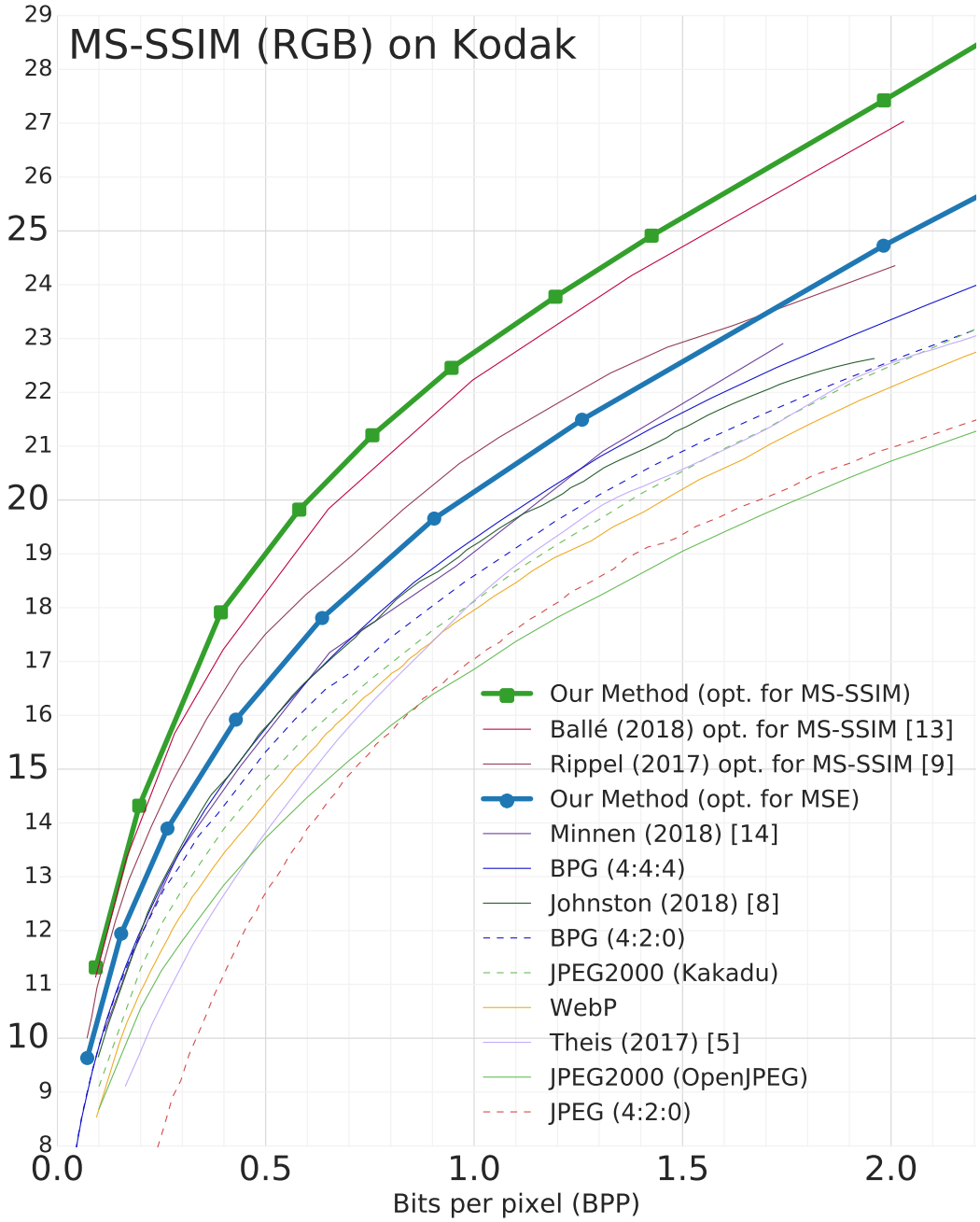


Figure 8: When trained for MS-SSIM, our method provides better rate-distortion performance compared to all of the baseline methods when evaluated on the Kodak image set using MS-SSIM. Note that when our method is trained for MSE, the MS-SSIM score is still competitive. Specifically, it has higher MS-SSIM scores than any of the standard codecs and all learned methods that aren't optimized specifically for MS-SSIM. To improve readability, this graph shows MS-SSIM scores in dB using the formula: $\text{MS-SSIM}_{\text{dB}} = -10 \log_{10}(1 - \text{MS-SSIM})$.

A.3 SSIM on Kodak

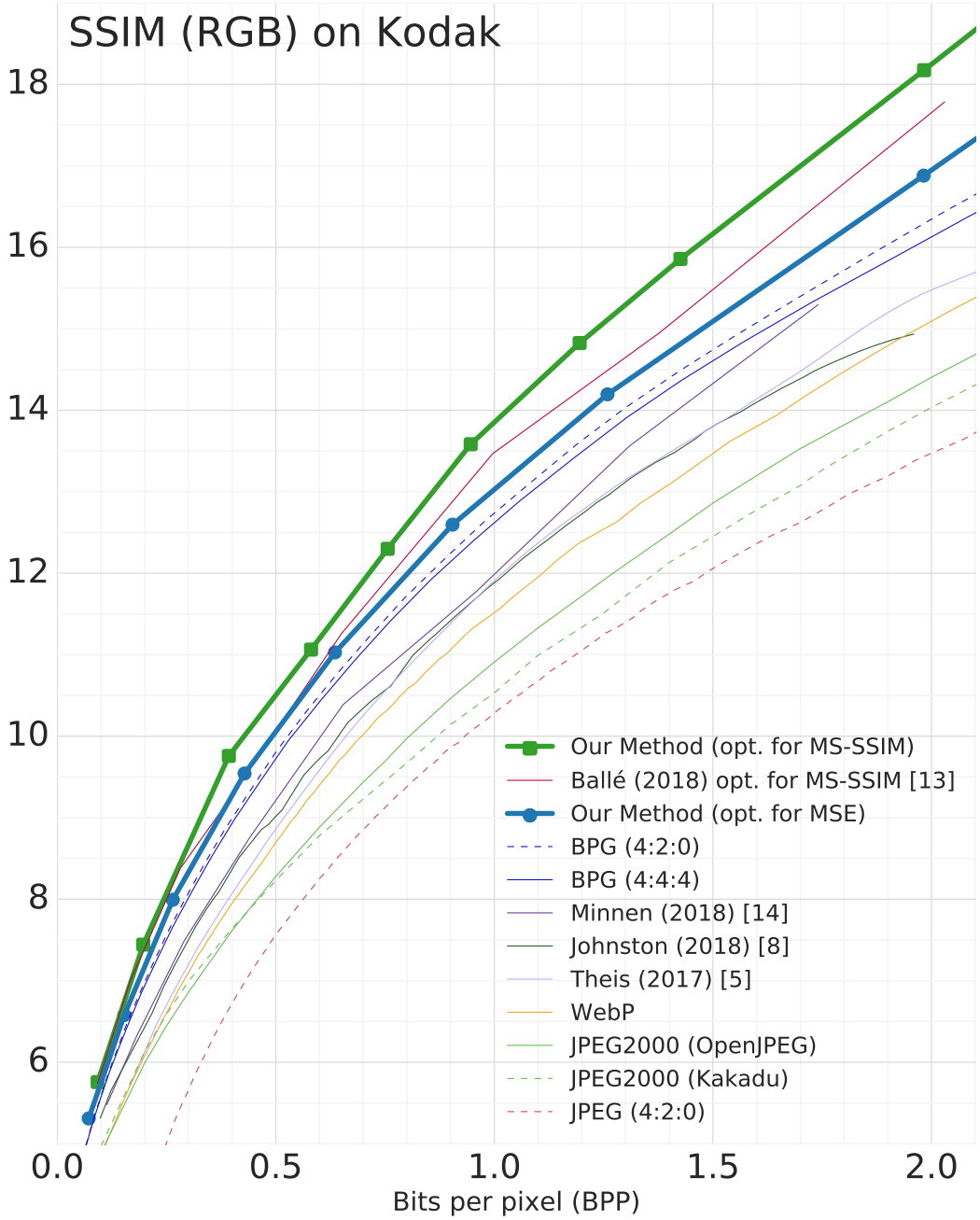


Figure 9: Our method performs very well according to SSIM when optimized for MS-SSIM or MSE. Both versions outperform all of the baseline methods, except for one inversion above 1.1 bpp where the MSE-optimized version is worse than Ballé et al. when that method is optimized for MS-SSIM. To improve readability, this graph shows SSIM scores in dB using the formula: $\mathrm{SSIM}_{\mathrm{dB}} = -10 \log_{10}(1 - \mathrm{SSIM})$.

A.4 PSNR on Tecnick

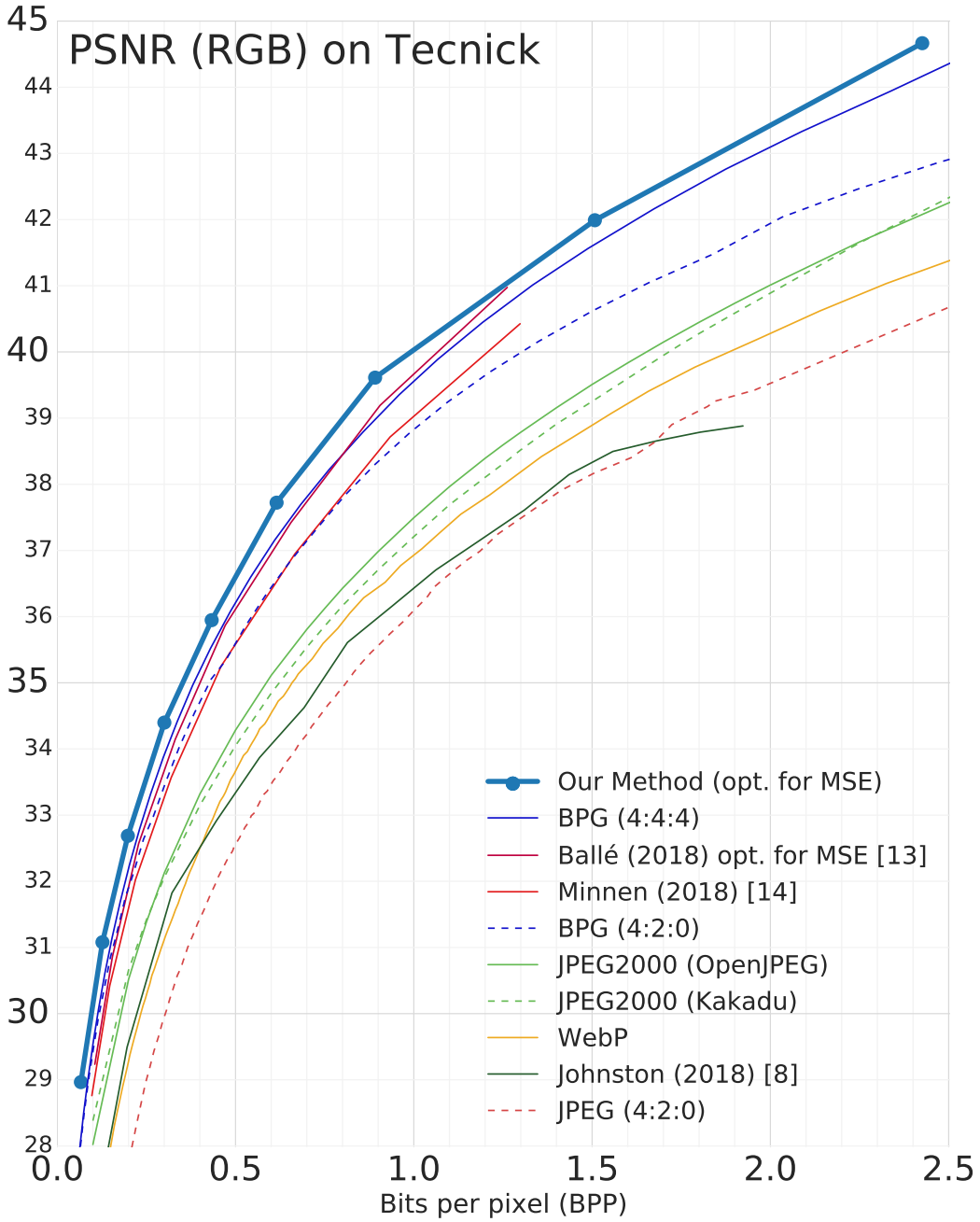


Figure 10: When trained for MSE, our method provides better rate-distortion performance compared to all of the baseline methods when evaluated on the Tecnick image set using PSNR. Each point on the RD curves is calculated by averaging over the PSNR and bit rate values for the 100 Tecnick images for a single Q value (for standard codecs) or λ (for learned methods). Each Tecnick image has resolution 1200×1200.

Appendix B Summary of Rate Savings

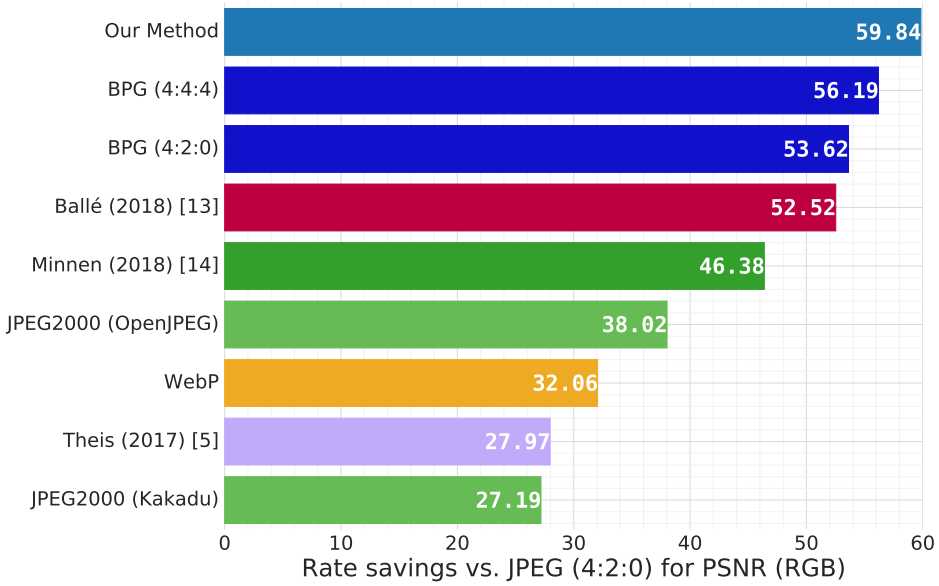


Figure 11: This graph shows the average rate savings of each method compared to JPEG (4:2:0) using PSNR on the Kodak image set. Higher scores correspond to larger rate savings compared to JPEG as the baseline codec. These scores are calculated from the RD graph for PSNR on Kodak (see Figure 7) over the shared PSNR range of 27.1–39.9 dB.

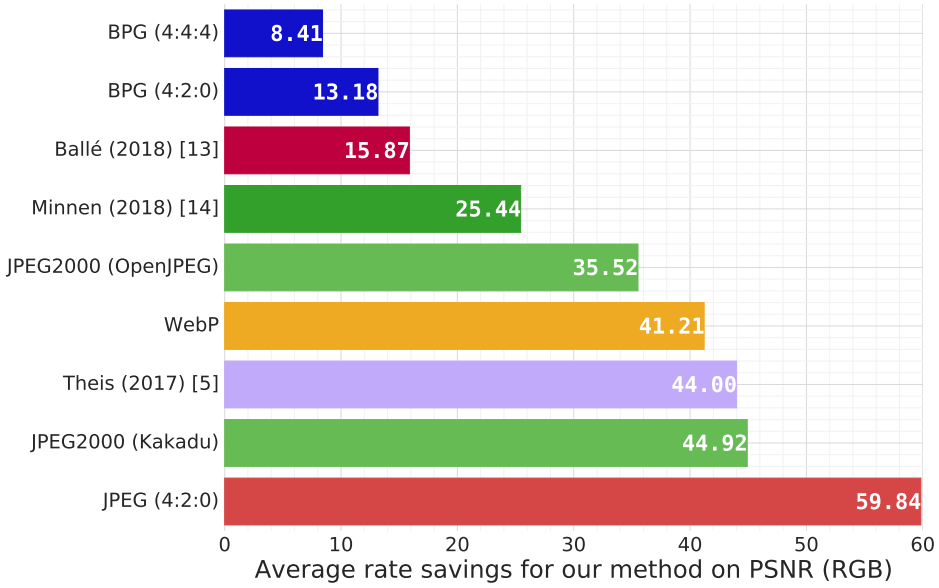


Figure 12: This graph shows the average rate savings of our model compared to other codecs. Larger values imply a larger savings, e.g., on average, our method saves 8.41% over BPG (4:4:4) and 35.52% over JPEG2000 (based on the OpenJPEG implementation). These scores are calculated from the RD graph for PSNR on Kodak (see Figure 7) over the shared PSNR range of 27.1–39.9 dB.

Appendix C Example Images

C.1 Kodak 15

(a) Our Method (0.1552 bpp)

(b) Scale-only (0.1788 bpp)

(c) BPG (0.1737 bpp)

(d) JPEG2000 (0.1557 bpp)

(e) JPEG (0.1634 bpp)

(f) Original

Figure 13: At similar bit rates, our method provides the highest visual quality on the Kodak 15 image. Note the geometric artifacts in BPG, e.g., on the girl's chin and cheek near the yellow face paint. JPEG2000 has severe artifacts and blurriness, while JPEG captures very little visual information at this bit rate. The reconstruction by the scale-only model is quite good, but it maintains less sharpness than our reconstruction (e.g., in the face paint and hair). Also note that both learning-based methods have a slight texture in the background, which is not in the original. We believe this arises due to the convolutional structure of the decoder network but is ultimately due to the use of mean squared error as the loss function, which is not very sensitive to subtle color shifts.

C.2 Kodak 19

(a) Our Method (0.2149 bpp)

(b) Scale-only (0.2205 bpp)

(c) BPG (0.2352 bpp)

(d) JPEG2000 (0.2152 bpp)

(e) JPEG (0.2309 bpp)

(f) Original

Figure 14: At similar bit rates, our method provides the highest visual quality on the Kodak 19 image. Note the aliasing in the fence in the scale-only version as well as a slight global color cast and blurriness in the yellow rope. BPG shows more "classical" compression artifacts, e.g., ringing around the top of the lighthouse and the roof of the middle building. BPG also introduces a few geometric artifacts in the sky, though it does preserve extra detail in the sky and fence (albeit with 9.4% more bits that our encoding). JPEG shows severe blocking artifacts at this bit rate, and JPEG2000 includes many artifacts in the sky and near object boundaries.

C.3 Kodak 20

(a) Our Method (0.2464 bpp)

(b) Scale-only (0.2760 bpp)

(c) BPG (0.2565 bpp)

(d) JPEG2000 (0.2499 bpp)

(e) JPEG (0.2521 bpp)

(f) Original

Figure 15: At similar bit rates, our method provides the highest visual quality on the Kodak 20 image. Note the artifacts in the sky in the BPG reconstruction and severe artifacts for JPEG and JPEG2000. The scale-only reconstruction is much closer visually, but our method has slightly more sharpness, e.g., in the red circles on the propeller and in the Yosemite Sam nose art, despite the fact that our method uses more than 10% fewer bits.

C.4 Kodak 23

(a) Our Method (0.1227 bpp)

(b) Scale-only (0.1378 bpp)

(c) BPG (0.1293 bpp)

(d) JPEG2000 (0.1241 bpp)

(e) JPEG (0.1576 bpp)

(f) Original

Figure 16: At similar bit rates, our method provides the highest visual quality on the Kodak 23 image. Note that the BPG reconstruction has some ringing and geometric artifacts (e.g., at the top of the red parrot's head), while JPEG2000 has many artifacts around object boundaries. The scale-only method has similar visual quality but is blurrier than our reconstruction, e.g., in the red parrot's eye and face.

Appendix D Architecture Comparison

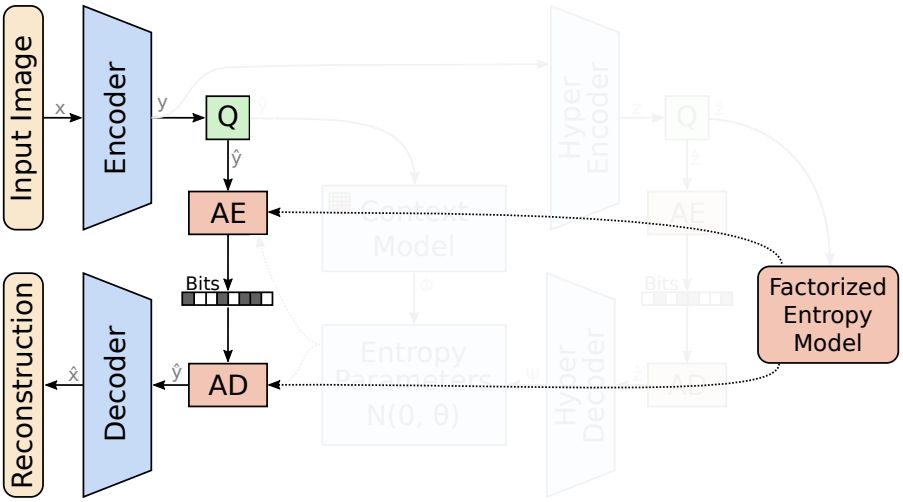


Figure 17: **Fully-Factorized Model** This model learns a fixed entropy model that is shared between the encoder and decoder systems. It is designed based on the assumption that all latents are i.i.d and ideally set to be a multinomial. In practice, it must be relaxed to ensure differentiability. The compression model is primarily an autoenocder composed of convolutional layers and nonlinear activations, where the main complication is due to ensuring that the entropy model is differentiable and that useful gradients flow through the quantized latent representation.

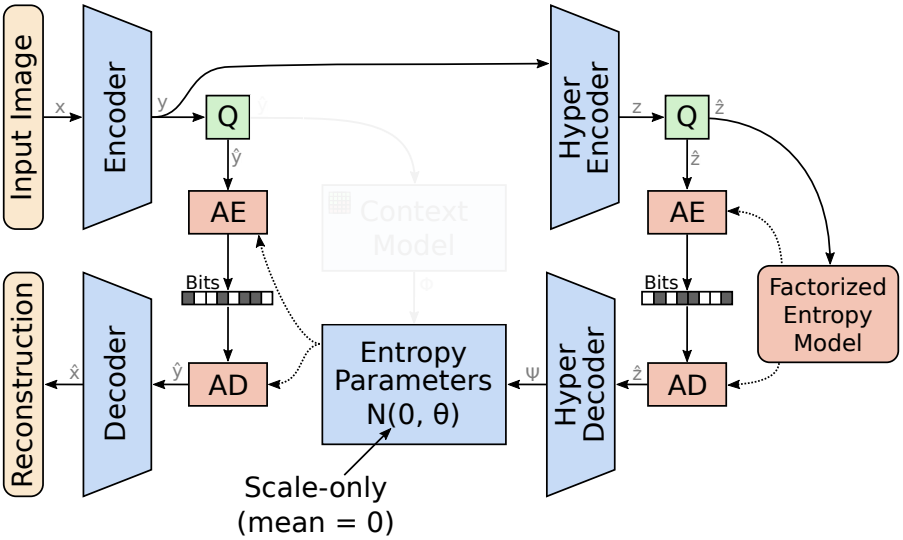


Figure 18: **Scale-only Hyperprior** This model uses a conditional Gaussian scale mixture (GSM) as the entropy model. The GSM is conditioned on a learned hyperprior, which is a (hyper-)latent representation formed by transforming the latents using the *Hyper-Encoder*. The *Hyper-Decoder* can then decode the hyperprior to create the scale parameters for the GSM. The main advantage of this model is that the entropy model is image-dependent and can be adapted for each individual code. The downside is that the compressed hyperprior must be transmitted with the compressed latents, which increases the total file size.

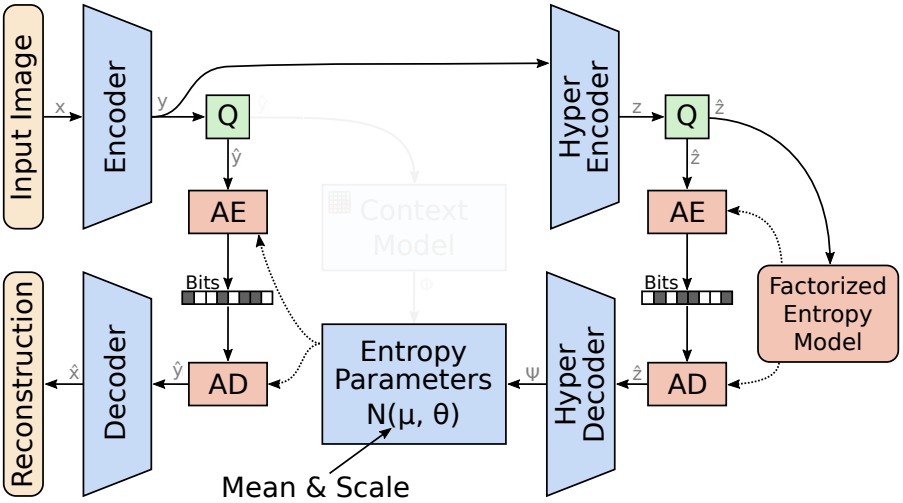


Figure 19: **Mean & Scale Hyperprior** This model variant is a simple extension of the scale-only hyperprior model shown in Figure 18 in which the GSM is replaced with a Gaussian mixture model (GMM). The *Hyper-Decoder* is therefore responsible for transforming the hyperprior into both the mean and scale parameters of the Gaussians.

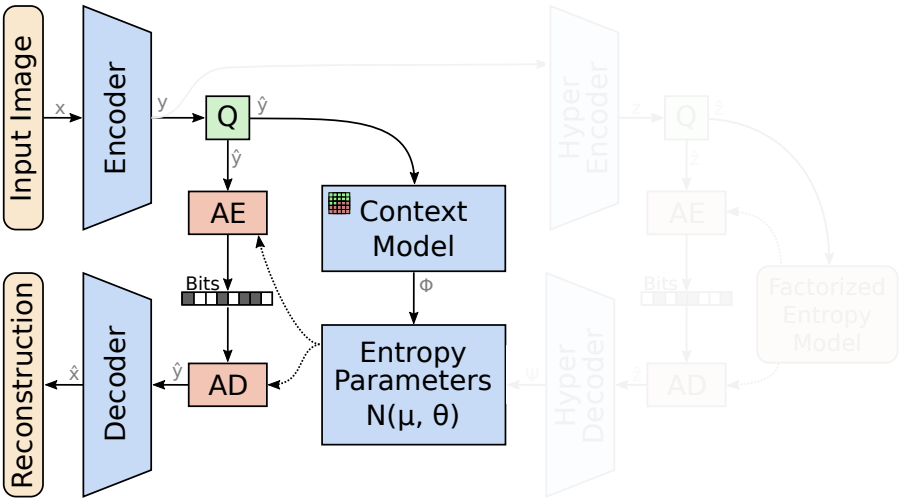


Figure 20: **Context-only Model** This model does not use a hyperprior and instead relies only on an autoregressive process to predict the parameters of the GMM-based entropy model. The benefit of this approach is that no additional bits are added to the bit-stream. The downside is that the autoregressive model can only access codes in its causal context since the decoder runs serially in raster-scan order.

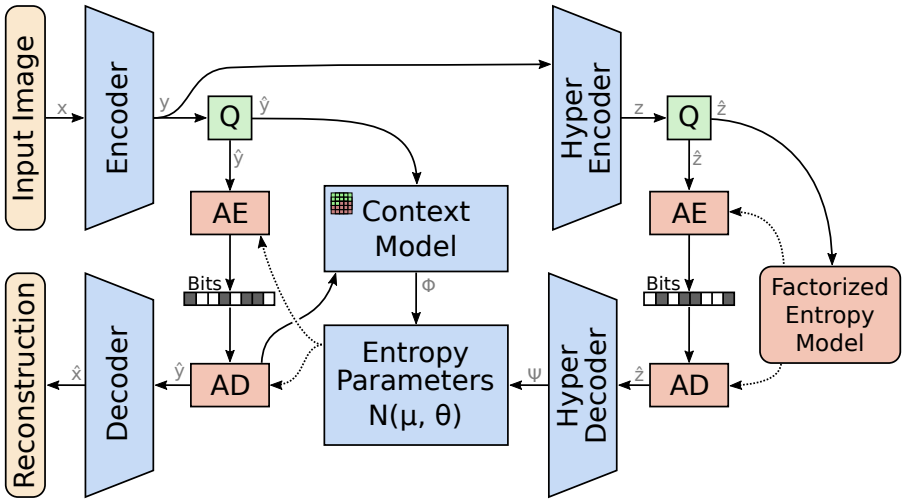


Figure 21: **Context + Hyperprior** By combining the mean & scale hyperprior with an autoregressive model, we form the full model presented in the paper. Our evaluation shows that the autoregressive model and the hyperprior are complementary, and that joint optimization leads to a compression model with better rate-distortion performance than either approach on its own.